TechRxiv
Powered by IEEE

Comparison of Three Recent Personalization Algorithms

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

18-11-2020 / 23-11-2020

CITATION

Shah, Shalin (2020): Comparison of Three Recent Personalization Algorithms. TechRxiv. Preprint.
https://doi.org/10.36227/techrxiv.13256384.v1

DOI

10.36227/techrxiv.13256384.v1

Comparison of Three Recent Personalization Algorithms

Venkataramana Kini {venkataramana.kini@target.com}
Shalin Shah {shalin.shah@target.com}

AI Sciences
Target Corporation
Sunnyvale, CA 94086, USA

Abstract

Personalization algorithms recommend products to users based on their previous interactions with the system. The products could be books, movies, or products in a retail system. The earliest personalization algorithms were based on factorization of the user-item matrix where each entry in the matrix would correspond to an interaction, or absence of an interaction of the user with the product. In this article, we compare three recently developed personalization algorithms. The three algorithms are Bayesian Personalized Ranking, Taxonomy Discovery for Personalized Recommendations and Multi-Matrix Factorization. We compare the three algorithms on the hit rate @ position 10 on a held out test set on 1 million users and 200 thousand items in the catalog of Target Corporation. We report our findings in table 1. We develop all three algorithms on an Apache Spark parallel implementation.

Keywords: recommender systems, bayesian personalized ranking, taxonomy discovery for personalized recommendations, multi-matrix factorization, apache spark

1 Introduction

Personalization systems are very important to the performance of a retailer in that it can improve the experience of a user by recommending items that the user would be interested in. The simplest and first personalization algorithm was based on factorizing the user-item matrix, also called matrix completion [1] [2]. These algorithms learn embeddings or latent factors of users and items and generate recommendations based on a similarity between the user and item latent factors. In an item-item recommender system, embeddings of items are learnt and recommendations are similar items to the item being viewed. The embeddings of items can be learnt through neural network based algorithms like

[3] or through graph based approaches like [4].

Bayesian Personalization Ranking (BPR) [5], Taxonomy Discovery for Personalized Recommendations (Taxonomy) [6] and Multi-Matrix Factorization (MMF) [7] are three algorithms that learn the latent factors of users and items based on their respective loss functions. We compare the three algorithms based on a hit rate and report our findings in table 1. We find that MMF performs the best of the three algorithms. The next section describes the three algorithms briefly.

2 Three Recommender Systems Algorithms

2.1 Taxonomy

Taxonomy Discovery for personalized recommendations [6] is work that learns the embeddings of users and items by alternating between latent factor updates and learning a taxonomy over the items based on the textual descriptions of the items. The algorithm can start with an existing taxonomy or it can learn a completely new taxonomy.

The algorithm also uses a BPR cost function.

The probability of preferring item i against item j is then:

$P(i > j|u) = \frac{exp(x_{ui})}{exp(x_{ui})+exp(x_{uj})}$

Where, $x_{ui} =< v_u, v_i > +q_i$

Where v_u is the user's latent factor and v_i and v_j are the item latent factors, where the user prefers item i to item j.

Each node and item share information with other nodes and items under the same parent in the hierarchy:

$v_i \sim N(v_{\pi_i}, \sigma^2 I)$

Where v_{π_i} is the latent factor of the parent of item i in the taxonomy.

q_i is a bias term which is modeled as a zero mean normal.

$q_i = N(0, \tau^2)$

The cost function to learn the latent factors is as follows:

$L_{uij} = log\sigma(x_{ui} - x_{uj}) - \sum_{k=1}^{L_i} \frac{||v_{i_k} - v_{i_{k+1}}||^2}{2\sigma^2} - \sum_{k=1}^{L_j} \frac{||v_{j_k} - v_{j_{k+1}}||^2}{2\sigma^2} - \frac{q_i}{\tau^2} - \frac{q_j}{\tau^2}$

The first term of the cost function is the sigmoid of the difference between the dot products of an interacted item against a non-interacted item. The second and third terms can be called regularization terms that ensure that nodes and parents have similar latent factors in the entire path in the taxonomy. The other two terms are bias terms that control for excessively popular items.

We implement the algorithm on Apache Spark as described in [8].

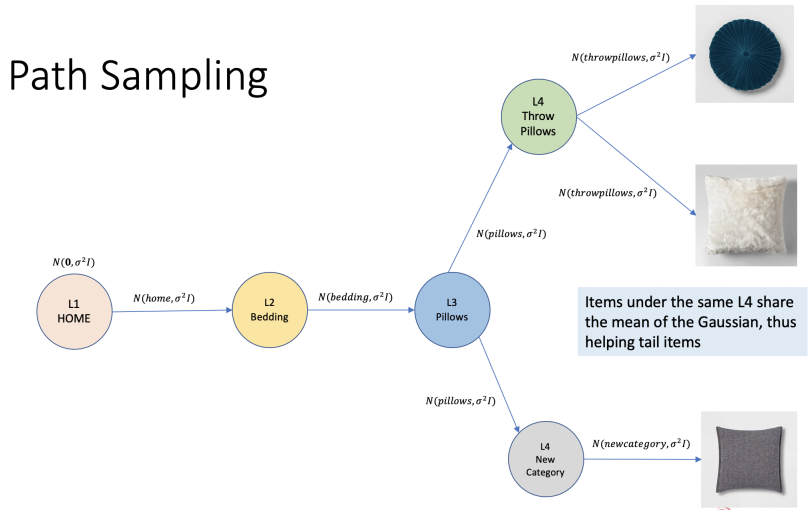

Figure 1: Gaussian Hierarchy in Taxonomy

2.2 BPR

The Bayesian Personalized Ranking method [5] uses a similar cost function:

The probability of a user, preferring item i against item j is:

$P(i > j|u) = \frac{exp(x_{ui})}{exp(x_{ui})+exp(x_{uj})}$

Where, $x_{ui} =< v_u, v_i > +q_i$

We also implement this on Apache Spark based on an implementation described in [9].

Table 1: Hit Rates on a 40% Test Set

Algorithm	Dimension	Test Set Hit Rate @10
MMF	50	9.6%
Taxonomy	50	4.97%
BPR	50	0.9%

2.3 MMF

The Multi-Matrix Factorization approach [7] learns latent factors of users and attributes. These attributes are properties of items like size, color etc. The algorithm also learns the preferences of users for attributes, and also how important each attribute is, for an item.

The predicted rating of an item is as follows:

$r_{ij} = \frac{1}{|M_j|} \sum_{k \in M_j} w_{ik} \theta_{jk} u_i^T f_k$

Where M_j is the set of attributes in an item, w_{ik} is the preference of the user i on the attribute k, θ_{jk} is the weight of attribute k in item j. u_i is the latent factor of user i and f_k is the latent factor of attribute k.

Similar to matrix factorization, this work uses the L_2 loss to optimize and learn these variables.

We implement this method on a distributed cluster using Apache Spark.

3 Results

We use a sample of 1 million users and 200 thousand items using 1 week of data from online interactions on Target.com. We hold a 40% of items as a test set to measure the hit rate. The results are in table 1. MMF clearly outperforms the other two algorithms. We developed a distributed implementation of all three algorithms on Apache Spark.

A hit is when an item preferred by a user occurs in the top 10 of all items sorted by the predicted ratings. The mean of the hits of all users is then the hit rate, which is shown in table 1. The hit rate is calculated on the held out test set.

4 Conclusion

In this article, we compared three recent algorithms for personalization. All three learn latent factors of the variables and use predicted ratings as a way of recommending items in the interests of a user. We implement all three algorithms on Apache Spark on a sample of 1 million users and 200 thousand items. We find that MMF clearly outperforms the other two algorithms. MMF has more parameters and learns the preferences of a user for attributes, rather than items. It then generates ratings for unobserved pairs of users and items based on an aggregate of all attributes of an item. Future work could be to run our algorithms on a much larger set of users and items and see if the results carry forward. Also, experiments could be conducted by increasing or decreasing the dimension of the latent factors (we use 50 for all three implementations).

References

[1] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[2] Shalin Shah. Introduction to matrix factorization for recommender systems.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[4] Shalin Shah and Venkataramana Kini. Hebbian graph embeddings. *arXiv preprint arXiv:1908.08037*, 2019.

[5] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[6] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 243–252, 2014.

[7] Yixin Su, Sarah Monazam Erfani, and Rui Zhang. Mmf: Attribute interpretable collaborative filtering. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[8] Shalin Shah, Ramasubbu Venkatesh, Jinghe Zhang, Venkataramana Kini, Sayon Majumdar, and Srikanth Ryali. Parallel taxonomy discovery. 2020.

[9] Alfredo Láinez Rodrigo and Luke de Oliveira. Distributed bayesian personalized ranking in spark.